LP can be a cure for Parameterized Problems

N.S. Narayanaswamy[1], Venkatesh Raman[2], M.S. Ramanujan[2], and Saket Saurabh[2]

1 **Department of Computer Science and Engineering, IIT Madras, Chennai, India.**
swamy@cse.iitm.ernet.in

2 **The Institute of Mathematical Sciences, Chennai 600113, India.**
{vraman|msramanujan|saket}@imsc.res.in

Abstract

We investigate the parameterized complexity of Vertex Cover parameterized above the optimum value of the linear programming (LP) relaxation of the integer linear programming formulation of the problem. By carefully analyzing the change in the LP value in the branching steps, we argue that even the most straightforward branching algorithm (after some preprocessing) results in an $O^*(2.6181^r)$ algorithm for the problem where r is the excess of the vertex cover size over the LP optimum. We write $O^*(f(k))$ for a time complexity of the form $O(f(k)n^{O(1)})$, where $f(k)$ grows exponentially with k.

Then, using known and new reductions, we give $O^*(2.6181^k)$ algorithms for the parameterized versions of Above Guarantee Vertex Cover, Odd Cycle Transversal, Split Vertex Deletion and Almost 2-SAT, and an $O^*(1.6181^k)$ algorithm for Kőnig Vertex Deletion, Vertex Cover Param by OCT and Vertex Cover Param by KVD. These algorithms significantly improve the best known bounds for these problems. The notable improvement is the bound for Odd Cycle Transversal for which this is the first major improvement after the first algorithm that showed it fixed-parameter tractable in 2003. We also observe that using our algorithm, one can obtain a simple kernel for the classical vertex cover problem with at most $2k - O(\log k)$ vertices.

1998 ACM Subject Classification G.2.2 Graph Theory, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Algorithms and data structures, Graph Algorithms, Parameterized Algorithms

Digital Object Identifier 10.4230/LIPIcs.STACS.2012.338

1 Introduction and Motivation

In this paper we revisit one of the most studied problems in parameterized complexity, the Vertex Cover problem. Given a graph $G = (V, E)$, a subset $S \subseteq V$ is called *vertex cover* if every edge in E has at least one end-point in S. The Vertex Cover problem is formally defined as follows.

29th Symposium on Theoretical Aspects of Computer Science (STACS'12).
Editors: Christoph Dürr, Thomas Wilke; pp. 338–349

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany


Vertex Cover	
Instance:	An undirected graph G and a positive integer k.
Parameter:	k.
Problem:	Does G have a vertex cover of of size at most k?

We start with a few basic definitions regarding parameterized complexity. For decision problems with input size n, and a parameter k, the goal in parameterized complexity is to design an algorithm with runtime $f(k)n^{O(1)}$ where f is a function of k alone, as contrasted with a trivial $n^{f(k)}$ algorithm. Problems which admit such algorithms are said to be fixed parameter tractable (FPT). The theory of parameterized complexity was developed by Downey and Fellows [6]. For recent developments, see the book by Flum and Grohe [7].

Vertex Cover was one of the earliest problems that was shown to be FPT [6]. After a long race, the current best algorithm for Vertex Cover runs in time $O(1.2738^k + kn)$ [3]. However, when $k < m$, the size of the maximum matching, the Vertex Cover problem is not interesting, as the answer is trivially NO. And if m is large (suppose, for example, the graph has a perfect matching), then for the cases the problem is interesting, the running time of the standard version is not practical, as k, in this case, is quite large. This led to the following natural "above guarantee version" of the Vertex Cover problem.

Above Guarantee Vertex Cover (agvc)	
Instance:	An undirected graph G, a maximum matching M and a positive integer ℓ.
Parameter:	ℓ.
Problem:	Does G have a vertex cover of of size at most $\|M\| + \ell$?

The agvc problem is not only a very natural parameterization of the classical Vertex Cover problem but is also very central in the "zoo" of parameterized problems. We refer to the Figure 1 for the details of problems reducing to agvc. This implies that an improved algorithm for this problem implies improved algorithm for several other problems, including Almost 2-SAT and Odd Cycle Transversal.

The first known parameterized algorithm for agvc was using a parameter preserving reduction to Almost 2-SAT. In Almost 2-SAT, we are given a 2-SAT formula ϕ, a positive integer k and the objective is to check whether there exists at most k clauses whose deletion from ϕ can make the resulting formula satisfiable. The Almost 2-SAT problem was introduced in [16] and a decade later it was shown by Razgon and Barry O'Sullivan [23] to have an $O^*(15^k)$ time algorithm, thereby proving fixed-parameter tractability of the problem when k is the parameter. In 2011, there were two new algorithms for the agvc problem [5, 22]. One using new structural results about König-Egerváry graphs — graphs where the size of a minimum vertex cover is equal to the size of a maximum matching [22] and the other by a novel reduction to an "above guarantee version" of the Multiway Cut problem [5]. The second algorithm runs in time $O^*(4^k)$ and this is the previously fastest known algorithm for agvc.

The algorithm presented in [5] for agvc is not only the fastest known algorithm but also differs from previous algorithms conceptually in that it introduces the concept of approaching problems above the guarantee obtained by solving the relaxation of linear programming. This novel approach, combined with the fact that an improvement on the above guarantee

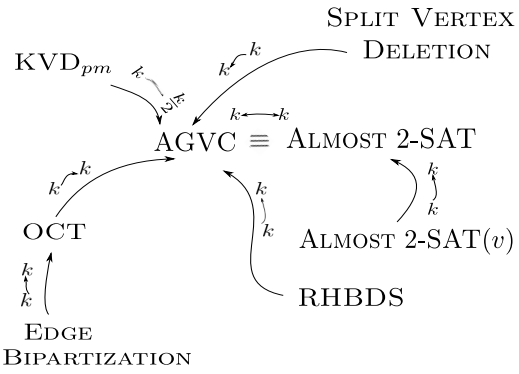

Figure 1 The zoo of problems around AGVC; An arrow from a problem P to a problem Q indicates that there is a parameterized reduction from P to Q with the parameter changes as indicated on the arrow.

versions of Vertex Cover improves the best known bounds for a number of parameterized problems, has motivated us to do a similar study for Vertex Cover.

The well known integer linear programming formulation (ILP) for Vertex Cover is as follows.

ILP formulation of Minimum Vertex Cover – ILPVC	
Instance:	A graph $G = (V, E)$.
Feasible Solution:	A function $x : V \to \{0, 1\}$ satisfying edge constraints $x(u) + x(v) \geq 1$ for each edge $(u, v) \in E$.
Goal:	To minimize $w(x) = \Sigma_{u \in V} x(u)$ over all feasible solutions x?

In the linear programming relaxation of the above ILP, the constraint $x(v) \in \{0, 1\}$ is replaced with $x(v) \geq 0$, for all $v \in V$. For a graph G, we call this relaxation LPVC(G). Clearly, every integer feasible solution is also a feasible solution to LPVC(G). If the minimum value of LPVC(G) is $vc^*(G)$ then clearly the size of a minimum vertex cover is at least $vc^*(G)$. So this leads to the following natural parameterization of Vertex Cover.

Vertex Cover above LP	
Instance:	An undirected graph G, positive integers k and $\lceil vc^*(G) \rceil$, where $vc^*(G)$ is the minimum value of LPVC(G)
Parameter:	$k - \lceil vc^*(G) \rceil$.
Problem:	Does G have a vertex cover of of size at most k?

Observe that since $vc^*(G) \geq m$, where m is the size of a maximum matching of G, we have that $k - vc^*(G) \leq k - m$. Thus any parameterized algorithm for Vertex Cover above LP is also an algorithm for AGVC and hence an algorithm for every problem described in Figure 1.

Our Results and Methodology. We develop an $O^*(2.6181^{(k-vc^*(G))})$ time for Vertex Cover above LP. Our algorithm is a simple branching algorithm. After a couple of preprocessing steps, the algorithm picks an arbitrary vertex v in the graph and recursively

Problem Name	Previous $f(k)$/Reference	New $f(k)$ in this paper
AGVC	4^k [5]	2.6181^k
Almost 2-SAT	4^k [5]	2.6181^k
RHorn-Backdoor Detection Set	4^k [5, 8]	2.6181^k
König Vertex Deletion	4^k [5, 18]	1.6181^k
Split Vertex Deletion	5^k [2]	2.6181^k
Odd Cycle Transversal	3^k [24]	2.6181^k
Vertex Cover Param by OCT	2^k (folklore)	1.6181^k
Vertex Cover Param by KVD	–	1.6181^k

Table 1 The table gives the previous $f(k)$ bound in the running time of various problems and the ones obtained in this paper.

tries to find a vertex cover of size at most k by considering whether v is in the solution or not. However, the analysis of the algorithm is more involved as it is not obvious that the measure $k - vc^*(G)$ will drop in the recursive steps. We string together several known results around linear programming relaxation of Vertex Cover to obtain this algorithm for Vertex Cover above LP. Some of the results we use are classical the Nemhauser-Trotter theorem and the properties of a "minimum surplus set". Using this algorithm we obtain an improved algorithm for every problem mentioned in Figure 1.

We give a list of problems with their previous best running time and the ones obtained in this paper in Table 1. The most notable one among them is the new algorithm for Odd Cycle Transversal, the problem of deleting at most k vertices to obtain a bipartite graph. The parameterized complexity of Odd Cycle Transversal was a long standing open problem in the area and only in 2003, Reed et al. [24], developed an algorithm for the problem running in time $O^*(3^k)$. In fact this was the first time that the iterative compression technique was used. However, there has been no further improvement over this algorithm in the last 9 years; though several reinterpretations of this algorithm have been published [9, 14].

We also find the algorithm for König Vertex Deletion, the problem of deleting at most k vertices to obtain a König graph very interesting. It is a natural generalization of the odd cycle transversal problem. In [18] it was shown that one can solve this problem by obtaining a minimum sized vertex cover of the graph and given a minimum vertex cover one can solve König Vertex Deletion in polynomial time. However, in this article we discover a relationship between the measure $k - vc^*(G)$ and the minimum number of vertices needed to delete to obtain a König graph, and this together with a reduction rule based on Nemhauser-Trotter theorem for König Vertex Deletion gives an algorithm with running time $O^*(1.6181^k)$.

We also note that using our algorithm, we obtain a simpler polynomial time algorithm for Vertex Cover that, given an input (G, k) returns an equivalent instance $(G' = (V', E'), k')$ such that $k' \leq k$ and $|V(G')| \leq 2k - c \log k$ for any fixed constant c. This is also known as a kernel for Vertex Cover in the literature. This improves the size bound on the previously known such algorithm [26], that gave an upper bound of $2k - c$ for any fixed constant c. Indepedently, Lampis [12] has also given a kernel for a Vertex Cover whose size is bounded by $2k - c \log k$.

We find this new algorithm for Odd Cycle Transversal and various other problems using an algorithm for Vertex Cover very exciting and hope that this will lead to a new race for Vertex Cover above LP like its classical counterpart Vertex Cover!

2 Preliminaries

Let $G = (V, E)$ denote a graph. For a subset S of V, the *subgraph of G induced by S* is denoted by $G[S]$ and it is defined as the subgraph of G with vertex set S and edge set $\{(u, v) \in E \ : \ u, v \in S\}$. By $N_G(u)$ we denote the (open) neighborhood of u, that is, the set of all vertices adjacent to u. Similarly, for a subset $T \subseteq V$, we define $N_G(T) = (\cup_{v \in T} N_G(v)) \setminus T$. When it is clear from the context, we drop the subscript G from the notation. The surplus of an independent set $X \subseteq V$ is defined as $\mathbf{surplus}(X) = |N(X)| - |X|$. The surplus of a graph G, $\mathbf{surplus}(G)$, is defined to be the minimum surplus over all independent sets in the graph.

By the phrase an *optimum solution* to LPVC(G), we mean a feasible solution with $x(v) \geq 0$ for all $v \in V$ minimizing the objective function $w(x) = \sum_{u \in V} x(u)$. It is well known that for any graph G, there exists an optimum solution to LPVC(G), such that $x(u) \in \{0, \frac{1}{2}, 1\}$ for all $u \in V$ [19]. Such a feasible optimum solution to LPVC(G) is called half integral and can be found in polynomial time [19]. In this paper we will always deal with half integral optimum solutions to LPVC(G). Thus by default whenever we will *say optimum solution* to LPVC(G) we will mean *half integral optimum solution* to LPVC(G). Let $VC(G)$ be the set of all minimum vertex covers of G and $vc(G)$ denote the size of a minimum vertex cover of G. Let $VC^*(G)$ be the set of all optimal solutions (including non half integral optimal solutions) to LPVC(G). By $vc^*(G)$ we denote the value of an optimum solution to LPVC(G). We define $V_i^x = \{u \in V : x(u) = i\}$ for each $i \in \{0, \frac{1}{2}, 1\}$ and define $x \equiv i$, $i \in \{0, \frac{1}{2}, 1\}$, if $x(u) = i$ for every $u \in V$. Clearly, $vc(G) \geq vc^*(G)$ and $vc^*(G) \leq \frac{|V|}{2}$ since $x = \frac{1}{2}$ is always a feasible solution to LPVC(G). We also refer to the $x \equiv \frac{1}{2}$ solution simply as the *all $\frac{1}{2}$ solution*. Proofs of results not appearing in the article will appear in the full version.

3 An Algorithm for Vertex Cover above LP

In this section we give an algorithm for Vertex Cover above LP. The algorithm has essentially two phases, a preprocessing phase and a branching phase. We first describe the preprocessing steps used in the algorithm and then give a simple description of the algorithm. Finally, we argue about its correctness and prove the desired running time bound on the algorithm.

3.1 Preprocessing

We describe two standard preprocessing rules to simplify the input instance. We first state the (known) results which allow for their correctness, and then describe the rules.

▶ **Lemma 1.** [20, 21] *For a graph G, in polynomial time, we can compute an optimal solution x to LPVC(G) such that all $\frac{1}{2}$ is the unique optimal solution to LPVC($G[V_{1/2}^x]$). Furthermore,* $\mathbf{surplus}(G[V_{1/2}^x]) > 0$.

▶ **Lemma 2.** [20] *Let G be a graph and x be an optimal solution to LPVC(G). There is a minimum vertex cover for G which contains all the vertices in V_1^x and none of the vertices in V_0^x.*

▶ Preprocessing Rule 1. Apply Lemma 1 to compute an optimal solution x to LPVC(G) such that all $\frac{1}{2}$ is the unique optimum solution to LPVC($G[V_{1/2}^x]$). If $V_0^x \cup V_1^x \neq \emptyset$ then delete the vertices in $V_0^x \cup V_1^x$ from the graph and reduce k by $|V_1^x|$.

The soundness/correctness of Preprocessing Rule 1 follows from Lemma 2. After the application of preprocessing rule 1, we know that $x \equiv \frac{1}{2}$ is the unique optimal solution to LPVC() of the resulting graph and the graph has a surplus of at least 1. This brings us to the next lemma which allows us to compute an independent set of a minimum surplus.

▶ **Lemma 3.** (see Theorem 6.1.4 in [15], see also [4] and [20]) *Given a graph G, the surplus of G, i.e. an independent set in G of minimum surplus, can be computed in polynomial time.*

▶ **Lemma 4.** [3, 20] *Let G be a graph, and let $Z \subseteq V(G)$ be an independent set such that* $\mathbf{surplus}(Z) = 1$ *and for every $Y \subseteq Z$,* $\mathbf{surplus}(Y) \geq \mathbf{surplus}(Z)$. *Then,*

1. *If the graph induced by $N(Z)$ is not an independent set, then there exists a minimum vertex cover in G that includes all of $N(Z)$ and excludes all of Z.*
2. *If the graph induced by $N(Z)$ is an independent set, let G' be the graph obtained from G by removing $Z \cup N(Z)$ and adding a vertex z, followed by making z adjacent to every vertex $v \in G \setminus (Z \cup N(Z))$ which was adjacent to a vertex in $N(Z)$ (also called* identifying *the vertices of $N(Z)$). Then, G has a vertex cover of size at most k if and only if G' has a vertex cover of size at most $k - |Z|$.*

▶ Preprocessing Rule 2. Using Lemma 3, find the minimum surplus independent set Z in G. If $\mathbf{surplus}(Z) = 1$, then apply Lemma 4 to reduce the instance. In other words, if the graph induced by $N(Z)$ is not an independent set, then include $N(Z)$ in the vertex cover, delete $Z \cup N(Z)$ from the graph, and decrease k by $|N(Z)|$ and otherwise, remove Z from the graph, identify the vertices of $N(Z)$, and decrease k by $|Z|$.

The soundness of Preprocessing Rule 2 follows from Lemma 4. As Z is a minimum surplus set in G, for every $Y \subseteq Z$, $\mathbf{surplus}\ (Y) \geq \mathbf{surplus}(Z)$.

After the exhaustive application of Preprocessing rules 1 and 2, for the resulting graph, all $\frac{1}{2}$ is the unique optimum solution to the LPVC() and the graph has a surplus of at least 2.

3.2 Branching

After the preprocessing rules are applied exhaustively until neither of the rules apply, we pick an arbitrary vertex u in the graph and branch on it. In other words, in one branch, we add u into the vertex cover, decrease k by 1, and delete u from the graph, and in the other branch, we add $N(u)$ into the vertex cover, decrease k by $|N(u)|$, and delete $\{u\} \cup N(u)$ from the graph. The correctness of this algorithm follows from the soundness of the preprocessing rules and the fact that the branching is exhaustive.

3.3 Analysis

In order to analyze the running time of our algorithm, we define a measure $\mu = \mu(G, k) = k - vc^*(G)$. We will first show that our preprocessing rules do not increase this measure. Following this, we will prove a lower bound on the decrease in the measure occurring as a result of the branching, thus allowing us to bound the running time of the algorithm in terms of the measure μ. For each case, we let (G', k') be the instance resulting by the application of the rule or branch, and let x' be an optimum solution to LPVC(G').

1. Consider the application of Preprocessing Rule 1. We know that $k' = k - |V_1^x|$. Since $x' \equiv \frac{1}{2}$ is the unique optimum solution to LPVC(G'), and G' comprises precisely the

vertices of $V^x_{1/2}$, the value of the optimum solution to LPVC(G') is exactly $|V^x_1|$ less than that of G. Hence, $\mu(G,k) = \mu(G',k')$.

2. We now consider the application of Preprocessing Rule 2.

(a) Suppose that $N(Z)$ was not independent. In this case, $k' = k - |N(Z)|$. We also know that $w(x') = \sum_{u \in V} x'(u) = w(x) - \frac{1}{2}(|Z| + |N(Z)|) + \frac{1}{2}(|V_1^{x'}| - |V_0^{x'}|)$. Adding and subtracting $\frac{1}{2}(|N(Z)|)$, we get $w(x') = w(x) - |N(Z)| - \frac{1}{2}(|Z| - |N(Z)|) + \frac{1}{2}(|V_1^{x'}| - |V_0^{x'}|)$. But, $Z \cup V_0^{x'}$ is an independent set in G, and $N(Z \cup V_0^{x'}) = N(Z) \cup V_1^{x'}$ in G. Since $\mathbf{surplus}(G) \geq 1$, $|N(Z \cup V_0^{x'})| - |Z \cup V_0^{x'}| \geq 1$. Hence, $w(x') = w(x) - |N(Z)| + \frac{1}{2}(|N(Z \cup V_0^{x'})| - |Z \cup V_0^{x'}|) \geq w(x) - |N(Z)| + \frac{1}{2}$. Thus, $\mu(G',k') \leq \mu(G,k) - \frac{1}{2}$.

(b) Suppose that $N(Z)$ was independent. In this case, $k' = k - |Z|$. We claim that $w(x') \geq w(x) - |Z|$. Suppose that this is not true. Then, it must be the case that $w(x') \leq w(x) - |Z| - \frac{1}{2}$. We will now consider three cases depending on the value $x'(z)$ where z is the vertex in G' resulting from the identification of $N(Z)$.

Case 1: $x'(z) = 1$. Now consider the following function $x'' : V \to \{0, \frac{1}{2}, 1\}$. For every vertex v in $G' \setminus \{z\}$, retain the value assigned by x', that is $x''(v) = x'(v)$. For every vertex in $N(Z)$, assign 1 and for every vertex in Z, assign 0. Clearly this is a feasible solution. But now, $w(x'') = w(x') - 1 + |N(Z)| = w(x') - 1 + (|Z| + 1) \leq w(x) - \frac{1}{2}$. Hence, we have a feasible solution of value less than the optimum, which is a contradiction.

Case 2: $x'(z) = 0$. Now consider the following function $x'' : V \to \{0, \frac{1}{2}, 1\}$. For every vertex v in $G' \setminus \{z\}$, retain the value assigned by x', that is $x''(v) = x'(v)$. For every vertex in Z, assign 1 and for every vertex in $N(Z)$, assign 0. Clearly this is a feasible solution. But now, $w(x'') = w(x') + |Z| \leq w(x) - \frac{1}{2}$. Hence, we have a feasible solution of value less than the optimum, which is a contradiction.

Case 3: $x'(z) = \frac{1}{2}$. Now consider the following function $x'' : V \to \{0, \frac{1}{2}, 1\}$. For every vertex v in $G' \setminus \{z\}$, retain the value assigned by x', that is $x''(v) = x'(v)$. For every vertex in $Z \cup N(Z)$, assign $\frac{1}{2}$. Clearly this is a feasible solution. But now, $w(x'') = w(x') - \frac{1}{2} + \frac{1}{2}(2|Z|) + \frac{1}{2} \leq w(x) - \frac{1}{2}$. Hence, we have a feasible solution of value less than the optimum, which is a contradiction.

Hence, $w(x') \geq w(x) - |Z|$, which implies that $\mu(G',k') \leq \mu(G,k)$.

3. We now consider the branching step.

a. Consider the case when we pick u in the vertex cover. In this case, $k' = k - 1$. We claim that $w(x') \geq w(x) - \frac{1}{2}$. Suppose that this is not the case. Then, it must be the case that $w(x') \leq w(x) - 1$. Consider the following assignment $x'' : V \to \{0, \frac{1}{2}, 1\}$ to LPVC(G). For every vertex $v \in V \setminus \{u\}$, set $x''(v) = x'(v)$ and set $x''(u) = 1$. Now, x'' is clearly a feasible solution and has a value at most that of x. But this contradicts our assumption that $x \equiv \frac{1}{2}$ is the unique optimum solution to LPVC(G). Hence, $w(x') \geq w(x) - \frac{1}{2}$, which implies that $\mu(G',k') \leq \mu(G,k) - \frac{1}{2}$.

b. Consider the case when we don't pick u in the vertex cover. In this case, $k' = k - |N(u)|$. We know that $w(x') = w(x) - \frac{1}{2}(|\{u\}| + |N(u)|) + \frac{1}{2}(|V_1^{x'}| - |V_0^{x'}|)$. Adding and subtracting $\frac{1}{2}(|N(u)|)$, we get $w(x') = w(x) - |N(u)| - \frac{1}{2}(|\{u\}| - |N(u)|) + \frac{1}{2}(|V_1^{x'}| - |V_0^{x'}|)$. But, $\{u\} \cup V_0^{x'}$ is an independent set in G, and $N(\{u\} \cup V_0^{x'}) = N(u) \cup V_1^{x'}$ in G. Since $\mathbf{surplus}(G) \geq 2$, $|N(\{u\} \cup V_0^{x'})| - |\{u\} \cup V_0^{x'}| \geq 2$. Hence, $w(x') = w(x) - |N(u)| + \frac{1}{2}(|N(\{u\} \cup V_0^{x'})| - |\{u\} \cup V_0^{x'}|) \geq w(x) - |N(u)| + 1$.

Hence, $\mu(G',k') \leq \mu(G,k) - 1$.

We have thus shown that the preprocessing rules do not increase the measure $\mu(G,k)$ and the branching step results in a $(\frac{1}{2},1)$ decrease in $\mu(G,k)=\mu$, resulting in the recurrence $T(\mu) \leq T(\mu-\frac{1}{2})+T(\mu-1)$ which solves to $(2.6181)^{\mu}=(2.6181)^{k-vc^*(G)}$. Thus we get a $(2.6181)^{(k-vc^*(G))}$ algorithm for Vertex Cover above LP.

▶ **Theorem 5.** Vertex Cover above LP *can be solved in time* $O^*((2.6181)^{k-vc^*(G)})$.

By applying the above theorem iteratively for increasing values of k, we can compute a minimum vertex cover of G and hence we have the following corollary.

▶ **Corollary 6.** *There is an algorithm that, given a graph G, computes a minimum vertex cover of G in time* $O^*(2.6181^{(vc(G)-vc^*(G))})$.

4 Applications

In this section we give several applications of the algorithm developed for Vertex Cover above LP.

4.1 An algorithm for Above Guarantee Vertex Cover

Since the value of the LP relaxation is at least the size of the maximum matching, our algorithm also runs in time $O^*(2.6181^{k-m})$ where k is the size of the minimum vertex cover and m is the size of the maximum matching.

▶ **Theorem 7.** Above Guarantee Vertex Cover *can be solved in time* $O^*(2.6181^{\ell})$ *time, where ℓ is the excess of the minimum vertex cover size above the size of the maximum matching.*

Now by the known reductions in [8, 17, 22] (see also Figure 1) we get the following corollary to Theorem 7.

▶ **Corollary 8.** Almost 2-SAT, Almost 2-SAT(v), RHorn-Backdoor Detection Set *can be solved in time* $O^*(2.6181^k)$. *However,* KVD_{pm} *can be solved in time* $O^*(1.6181^k)$.

4.2 Algorithms for Odd Cycle Transversal and Split Vertex Deletion

We describe a generic algorithm for both Odd Cycle Transversal and Split Vertex Deletion. Let $X, Y \in \{\text{Clique, Independent Set}\}$. A graph G is called an (X,Y)-*graph* if its vertices can be partitioned into X and Y. Observe that when X and Y are both *independent set*, this corresponds to a *bipartite graph* and when X is *clique* and Y is *independent set*, this corresponds to a *split graph*. In this section we outline an algorithm that runs in time $O^*(2.6181^k)$ and solves the following problem.

(X,Y)-Transversal Set

Instance:	An undirected graph G and a positive integer k.
Parameter:	k.
Problem:	Does G have a vertex subset S of size at most k such that its deletion leaves a (X,Y)-graph?

We solve the (X,Y)-Transversal Set problem by using a reduction to AGVC that takes k to k [25].

▶ **Theorem 9.** (X,Y)-Transversal Set *can be solved in time* $O^*(2.6181^k)$.

As a corollary to the above theorem we get the following new results.

▶ **Corollary 10.** Odd Cycle Transversal *and* Split Vertex Deletion *can be solved in time* $O^*(2.6181^k)$.

4.3 An algorithm for König Vertex Deletion

A graph G is called König if the size of a minimum vertex cover equals that of a maximum matching in the graph. Clearly bipartite graphs are König but there are non-bipartite graphs that are König (a triangle with an edge attached to one of its vertices, for example). The König Vertex Deletion problem is stated as follows.

König Vertex Deletion (KVD)	
Instance:	An undirected graph G and a positive integer k.
Parameter:	k.
Problem:	Does G have a vertex subset S of size at most k such that $G \setminus S$ is a König graph?

This problem is a natural generalization of the Odd Cycle Transversal problem. If the input graph G to König Vertex Deletion has a perfect matching then this problem is called KVD_{pm}. By Corollary 8, we already know that KVD_{pm} has an algorithm with running time $O^*(1.6181^k)$ by a polynomial time reduction to AGVC, that takes k to $k/2$. However, there is no known reduction if we do not assume that the input graph has a perfect matching and it required several interesting structural theorems in [18] to show that KVD can be solved as fast as AGVC. Here, we outline an algorithm for KVD that runs in $O^*(1.6181^k)$ and uses an interesting reduction rule. However, for our algorithm we take a slight detour and solve a slightly different, although equally interesting problem. Given a graph, a set S of vertices is called *König vertex deletion set (kvd set)* if its removal leaves a König graph. The auxiliary problem we study is following.

Vertex Cover Param by KVD	
Instance:	An undirected graph G, a König vertex deletion set S of size at most k and a positive integer ℓ.
Parameter:	k.
Problem:	Does G have a vertex cover of size at most ℓ?

This fits into the recent study of problems parameterized by other structural parameters. See, for example Odd Cycle Transversal parameterized by various structural parameters [11] or Treewidth parameterized by vertex cover [1] or Vertex Cover parameterized by feedback vertex set [10].

For our proofs we will use the following characterization of König graphs.

▶ **Lemma 11.** [18, Lemma 1] *A graph* $G = (V, E)$ *is König if and only if there exists a bipartition of* V *into* $V_1 \uplus V_2$*, with* V_1 *a vertex cover of* G *such that there exists a matching across the cut* (V_1, V_2) *saturating every vertex of* V_1.

Note that in Vertex Cover param by KVD, $G \setminus S$ is a König graph. So one could branch on all subsets of S to include in the output vertex cover, and for those elements not picked in S, we could pick its neighbors in $G \setminus S$ and delete them. However, the resulting graph need not be König adding to the complications. Note, however, that such an algorithm would yield an $O^*(2^k)$ algorithm for Vertex Cover Param by OCT. That is, if S were an odd cycle transversal then the resulting graph after deleting the neighbors of vertices not picked from S will remain a bipartite graph, where an optimum vertex cover can be found in polynomial time.

Given a graph $G = (V, E)$ and two disjoint vertex subsets V_1, V_2 of V, we let (V_1, V_2) denote the bipartite graph with vertex set $V_1 \cup V_2$ and edge set $\{\{u, v\} : \{u, v\} \in E \text{ and } u \in V_1, v \in V_2\}$. Now, we describe an algorithm based on Theorem 5, that solves Vertex Cover param by KVD in time $O^*(1.6181^k)$.

▶ **Theorem 12.** Vertex Cover Param by KVD *can be solved in time* $O^*(1.6181^k)$.

Proof. Let G be the input graph, S be a kvd set of size at most k. We first apply Lemma 1 on $G = (V, E)$ and obtain an optimum solution to LPVC(G) such that all $\frac{1}{2}$ is the unique optimum solution to LPVC($G[V^x_{1/2}]$). Due to Lemma 2, this implies that there exists a minimum vertex cover of G that contains all the vertices in V^x_1 and none of the vertices in V^x_0. Hence, the problem reduces to finding a vertex cover of size $\ell' = \ell - |H|$ for the graph $G' = G[V^x_{1/2}]$. Before we describe the rest of the algorithm, we prove the following lemma regarding kvd sets in G and G' which shows that if G has a kvd set of size at most k then so does G'. Even though this looks straight forward, the fact that König graphs are not hereditary (i.e. induced subgraphs of König graphs need not be König) makes this a non-trivial claim to prove.

▶ **Lemma 13.** *Let G and G' be defined as above. Let S be a kvd set of graph G of size at most k. Then, there is a kvd set of graph G' of size at most k.*

We now show that $\mu = vc(G') - vc^*(G') \leq \frac{k}{2}$. Let O be a kvd set of G' and define G'' as the Kónig graph $G' \setminus O$. We know that $|M| = vc(G'') = vc^*(G'')$, where M is a maximum matching in the graph G''. This implies that $vc(G') \leq vc(G'') + |O| = |M| + |O|$. But, we also know that $vc^*(G') \geq |M| + \frac{1}{2}(|O|)$ and hence, $vc(G') - vc^*(G') \leq \frac{1}{2}(|O|)$. By Lemma 13, we know that there is an O such that $|O| \leq k$ and hence, $vc(G') - vc^*(G') \leq \frac{k}{2}$.

By Corollary 6, we can find a minimum vertex cover of G' in time $O^*(2.6181^{vc(G')-vc^*(G')})$ and hence in time $O^*(2.6181^{k/2})$. If the size of the minimum vertex cover obtained for G' is at most ℓ', then we return yes else we return no. This completes the proof of the theorem. ◀

It is known that, given a minimum vertex cover, a minimum sized kvd set can be computed in polynomial time [18]. Hence, Theorem 12 has the following corollary.

▶ **Corollary 14.** KVD *can be solved in time* $O^*(1.6181^k)$.

Since the size of a minimum Odd Cycle Transversal is at least the size of a minimum Konig Vertex Deletion set, we also have the following corollary.

▶ **Corollary 15.** Vertex Cover Param by OCT *can be solved in time* $O^*(1.6181^k)$.

4.4 An improved kernel for Vertex Cover

We give a kernelization for Vertex Cover based on Theorem 5 as follows. Exhaustively, apply the Preprocessing rules 1 and 2 (see Section 3). When the rules no longer apply, if $k - vc^*(G) \leq \log k$, then solve the problem in time $O^*(2.6181^{\log k}) = O(n^{O(1)})$. Otherwise, just return the instance. We claim that the number of vertices in the returned instance is at most $2k - 2\log k$. Since $k - vc^*(G) > \log k$, $vc^*(G)$ is upper bounded by $k - \log k$. But, we also know that when Preprocessing Rule 1 is no longer applicable, all $\frac{1}{2}$ is the unique optimum to LPVC(G) and hence, the number of vertices in the graph G is twice the value of the optimum value of LPVC(G). Hence, $|V| = 2vc^*(G) \leq 2(k - \log k)$. Observe that by the same method we can also show that in the reduced instance the number of vertices is upper bounded by $2k - c\log k$ for any fixed constant c. Indepedently, Lampis [12] has also given a kernel for a Vertex Cover whose size is bounded by $2k - c\log k$.

5 Conclusion and Further Work

We have demonstrated that using the drop in LP values to analyze branching algorithms can give powerful results for parameterized complexity. Recently, in [13], a significantly faster algorithm for Vertex Cover above LP, running in time $O^*(2.3146^k)$, has been obtained. We believe that our algorithm is the beginning of a race to improve the running time bound for agvc and possibly for the classical vertex cover problem, for which there has been no progress in the last several years after an initial plethora of results.

Our other contribution is to exhibit several parameterized problems that are equivalent to or reduce to agvc through parameterized reductions. We observe that as the parameter change in these reductions are linear, any upper or lower bound results for kernels for one problem will carry over for the other problems too (subject to the directions of the reductions).

References

1 H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Preprocessing for treewidth: A combinatorial analysis through kernelization. In L. Aceto, M. Henzinger, and J. Sgall, editors, *ICALP (1)*, volume 6755 of *Lecture Notes in Computer Science*, pages 437–448. Springer, 2011.

2 L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett.*, 58(4):171–176, 1996.

3 J. Chen, I. A. Kanj, and G. Xia. Improved upper bounds for vertex cover. *Theor. Comput. Sci.*, 411(40-42):3736–3756, 2010.

4 M. Chlebík and J. Chlebíková. Crown reductions for the minimum weighted vertex cover problem. *Discrete Applied Mathematics*, 156(3):292–312, 2008.

5 M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk. On multiway cut parameterized above lower bounds. In *IPEC*, 2011.

6 R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer-Verlag, New York, 1999.

7 J. Flum and M. Grohe. *Parameterized Complexity Theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2006.

8 G. Gottlob and S. Szeider. Fixed-parameter algorithms for artificial intelligence, constraint satisfaction and database problems. *Comput. J.*, 51(3):303–325, 2008.

9 F. Hüffner. Algorithm engineering for optimal graph bipartization. *J. Graph Algorithms Appl.*, 13(2):77–98, 2009.

10 B. M. P. Jansen and H. L. Bodlaender. Vertex cover kernelization revisited: Upper and lower bounds for a refined parameter. In T. Schwentick and C. Dürr, editors, *STACS*, volume 9 of *LIPIcs*, pages 177–188. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

11 B. M. P. Jansen and S. Kratsch. On polynomial kernels for structural parameterizations of odd cycle transversal. *CoRR*, abs/1107.3658, To appear in IPEC 2011.

12 M. Lampis. A kernel of order $2k - c \log k$ for vertex cover. *Inf. Process. Lett.*, 111 (23-24):1089–1091, 2011.

13 D. Lokshtanov, N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, and S. Saurabh. Improved Parameterized Algorithms using LP. *Manuscript*, 2012.

14 D. Lokshtanov, S. Saurabh, and S. Sikdar. Simpler parameterized algorithm for oct. In J. Fiala, J. Kratochvíl, and M. Miller, editors, *IWOCA*, volume 5874 of *Lecture Notes in Computer Science*, pages 380–384. Springer, 2009.

15 L. Lovász and M. D. Plummer. *Matching Theory*. North Holland, Oxford, 1986.

16 M. Mahajan and V. Raman. Parameterizing above guaranteed values: Maxsat and maxcut. *J. Algorithms*, 31(2):335–354, 1999.

17 D. Marx and I. Razgon. Constant ratio fixed-parameter approximation of the edge multicut problem. *Inf. Process. Lett.*, 109(20):1161–1166, 2009.

18 S. Mishra, V. Raman, S. Saurabh, S. Sikdar, and C. R. Subramanian. The complexity of könig subgraph problems and above-guarantee vertex cover. *Algorithmica*, 58, 2010.

19 G. L. Nemhauser and L. E. Trotter. Properties of vertex packing and independence system polyhedra. *Mathematical Programming*, 6:48–61, 1974. 10.1007/BF01580222.

20 G. L. Nemhauser and L. E. Trotter. Vertex packings: Structural properties and algorithms. *Mathematical Programming*, 8:232–248, 1975. 10.1007/BF01580444.

21 J.-C. Picard and M. Queyranne. On the integer-valued variables in the linear vertex packing problem. *Mathematical Programming*, 12(1):97–101, 1977.

22 V. Raman, M. S. Ramanujan, and S. Saurabh. Paths, flowers and vertex cover. In C. Demetrescu and M. Halldórsson, editors, *Algorithms – ESA 2011*, volume 6942 of *Lecture Notes in Computer Science*, pages 382–393. Springer Berlin / Heidelberg, 2011.

23 I. Razgon and B. O'Sullivan. Almost 2-sat is fixed-parameter tractable. *J. Comput. Syst. Sci.*, 75(8):435–450, 2009.

24 B. A. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004.

25 S. Saurabh. *Exact Algorithms for some parameterized and optimization problems on graphs*. PhD thesis, 2008.

26 A. Soleimanfallah and A. Yeo. A kernel of order $2k - c$ for vertex cover. *Discrete Mathematics*, 311(10-11):892–895, 2011.